**DOCUMENT RESUME**

ED 332 695 IR 015 082

AUTHOR Spector, J. Michael; And Others
TITLE Modeling User Interactions with Instructional Design Software.
PUB DATE Apr 91
NOTE 22p.; Paper presented at the Annual Meeting of the American Educational Research Association (Chicago, IL, April 3-7, 1991).
PUB TYPE Reports - Evaluative/Feasibility (142) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Authoring Aids (Programing); Computer Assisted Instruction; *Computer Software Development; Computer Software Evaluation; Courseware; *Instructional Design; Instructional Effectiveness; *Interaction; *Military Personnel; Military Training; *Models; Postsecondary Education; Technical Education; User Needs (Information)
IDENTIFIERS *Transaction Shells

ABSTRACT

As one of a series of studies being conducted to develop a useful (predictive) model of the instructional design process that is appropriate to military technical training settings, this study performed initial evaluations on two pieces of instructional design software developed by M. David Merrill and colleagues at Utah State University i.e., Merrill's Naming Transaction Shell and Merrill's Checklist Procedures Transaction Shell. The software provided the capability to design, develop, and deliver computer-based instruction for teaching both the parts of a device and simple checklist procedures pertaining to the device. The primary objective of the study was to identify the relevant variables to be used in determining the acceptability, the generalizability, and the effectiveness of the instructional design software. In an initial study, the Naming Transaction Shell was evaluated by a subject matter expert with no previous computer experience who was an instructor at the U.S. Air Force Academy. A second study conducted with six subject matter experts and the same instructor used two transaction shells--naming the parts and checklist procedures. In each study, subjects had about 30 hours to design and develop a lesson module for one hour of student instruction. In the second study, a simple drawing program was used to create graphics. Both studies indicate that the transaction shells provided a highly cost-effective authoring environment and that this approach has good potential for courseware authoring. (SLD)

# MODELING USER INTERACTIONS WITH INSTRUCTIONAL DESIGN SOFTWARE

J. Michael Spector

Visiting Research Associate
Air Force Armstrong Laboratory
AL/HRTC, Brooks AFB, TX 78235-5601

&

Daniel J. Muraida

AIDA Project Manager
Air Force Armstrong Laboratory
AL/HRTC, Brooks AFB, TX 78235-5601

&

Mary R. Marlino

Director of Educational Technology
United States Air Force Academy
HQ USAFA/DFTE, USAFA, CO 80840-5000

Presented at the 1991 Annual Meeting of the American Educational Research Association, Chicago, IL (Military Training Special Interest Group).

The opinions expressed herein are those of the authors and do not necessarily reflect the views of the United States Air Force.

Keywords: automated instructional design, computer-based instruction (CBI), instructional systems development (ISD), modeling instructional design, soft modeling.

## ABSTRACT

Many researchers are attempting to develop automated instructional design systems to guide subject matter experts through the courseware authoring process (Merrill, 1990). What is lacking in a number of existing research and development efforts, however, is a systematic method for analyzing the interplay between user characteristics, the authoring environment, and the resulting quality of computer-based instruction (CBI). The Air Force Armstrong Laboratory (Human Resources Directorate) is funding a project called the Advanced Instructional Design Advisor (AIDA) and has taken on the task of analyzing the relevant variables involved in authoring CBI in military settings in order to create an effective instructional design advisor.

This paper is a discussion of the initial evaluations of two pieces of instructional design software developed by M. David Merrill and colleagues at Utah State University. The software provided the capability to design, develop, and deliver computer-based instruction for teaching: 1) the parts of a device (names, locations, and functions), and 2) simple checklist procedures pertaining to those devices. Merrill provided the software as part of a letter of agreement with the Armstrong Laboratory.

The primary objective of this study was to identify the relevant factors to be used in determining the acceptability, the generalizability, the executability, and the effectiveness of the aforementioned instructional design software. The overall goal of this series of studies is to develop a useful (predictive) model of the instructional design process that is appropriate to military technical training settings.

## Introduction

The purpose of the instructional design process is to structure the environment to provide a learner with conditions which will foster learning (Gagné, Briggs, and Wager, 1988). When learning goals are simple and delivery media are restricted to lecture and blackboard, this process is manageable. However, as learning goals grow in complexity and media choices proliferate, the complexity of the instructional design process generates a number of as yet unsolved problems.

The stages and associated procedures of the instructional design process have been described by a number of authors (e.g., Gagné et al., 1988; Dick and Carey, 1985; Tennyson, 1989). The description typically occurs at the task level in terms of identifying goals and learner characteristics, analyzing instructional requirements, writing objectives, developing test items, instructional strategies, and instructional materials, and conducting formative and summative evaluations. While these procedures are part and parcel of instructional design, many fledgling practitioners tend to implement them at a superficial level or in a rigidly linear fashion (Spector, Muraida, and Dallman, 1990).

The reality of the instructional design process is that it is a highly complex and ill-formed problem solving process (Duchastel, 1990; Pirolli, 1989). Successful instructional designs require that the designer possess or have access to

specific knowledge about what works for the learners in question as well as methods for linking student experiences with the cognitive processes involved in learning. Much of the information in the former category can be acquired only with broad and extensive experience. Most of the information in the latter category has yet to filter down from cognitive science to the repertoire of the average instructional designer. There are three implications of this situation:

1) Becoming a fully competent instructional designer requires a relatively long developmental period, and

2) There is no general method of guaranteeing that optimal designs will be implemented and delivered.

3) Cost considerations will be a concern in most CBI development efforts.

## Problem

### Background

The problem of automated instructional design for computer-based instruction (CBI) presents a number of additional complications for the designer. The designer has to be cognizant of the capabilities and the limitations of the computer and its

peripheral input/output devices as a delivery medium. In addition, the designer must also be able to exploit the unique capabilities of the computer as a rapid instructional development prototyping medium in order to contain the cost of CBI development.

When instructional delivery is targeted for a computer, it is nearly irresistible to make the computer an integral part of the design and development of the same instruction. Automated tools for instructional design, however, may inadvertently add to the difficulty of the process they are intended to simplify. Recent studies of design tasks in areas such as architecture and mechanics have led instructional theorists to postulate a general framework for analyzing the process of design (Goel and Pirolli, 1989). Goel and Pirolli assert that many of the problems characteristic of instructional design have their analogues in other domains.

If automated tools are to realize their potential to augment the designer's problem solving capacity, then it will be necessary to create models of the instructional design process at a level of granularity that permits researchers to relate critical factors in terms of cognitive complexity of the design task, scope of the design effort, design experience, and authoring system flexibility with respect to various indices of courseware quality (e.g., level of interactivity). It is necessary to develop models which accurately reflect the instructional design process from the perspective of the user --

the person who has the task of designing and developing the instruction.

Models of Instructional Design

Models of instructional design and models of design in other domains undeniably exist. The most prevalent instructional design models are based on an engineering approach to curriculum development called Instructional Systems Development (ISD). There are a number of these models in use today (Andrews and Goodson, 1980; Tennyson, 1989). They typically divide the process into five stages and prescribe more or less detailed procedures for each stage. A partial ISD model is depicted in the table below:

| ISD PHASE | TYPICAL GOALS |
|---|---|
| ANALYSIS | Define training requirements.<br>Analyze target population.<br>Establish performance levels. |
| DESIGN | Specify instructional objectives.<br>Group and sequence objectives.<br>Specify evaluation system. |
| DEVELOPMENT | Develop learning activities.<br>Develop test items.<br>Perform formative evaluation. |
| IMPLEMENTATION | Implement learning activities.<br>Administer test items.<br>Assess student results. |
| MAINTENANCE | Revise content materials.<br>Revise test items.<br>Assess course effectiveness. |

Table 1. Typical ISD Model

## Models of Courseware Authoring

In arguing for an updated ISD model appropriate to CBI settings, Tennyson astutely observed that there is a need to tend to the specific authoring activities in each phase (1989). In the course of describing specific courseware authoring activities, Tennyson has built a potentially more useful model of the instructional design process. The usefulness of Tennyson's model results from its emphasis on human activities (what instructional designers do) as opposed to the idealized results of those human activities.

Making the instructional designer a primary unit of analysis in the model immediately introduces an additional complicating factor: levels of experience. What the experienced designer does may differ radically (in both order and substance) from what the novice designer may do, yet they may both produce effective (or ineffective) courseware designs. It is a commonplace in cognitive psychology that experts perform differently than novices (e.g., Glaser, 1989). Experts achieve levels of automaticity with regard to common procedures; evidence also indicates that they chunk problems much differently than novices.

In addition to the need to account for relevant instructional design experiences, a complete model of the computer-based instructional design process will need to take into account how and whether various activities are supported within a particular automated instructional design system. For example, if the instructional design model identifies as a

critical authoring activity the specification of an instructional strategy, then relevant questions to ask of a candidate courseware authoring tool are the following:

1) Does the system allow users to input strategies?
2) If so, is strategy input optional or required?
3) If optional, how is a default strategy determined?
4) What does the system do with the user's input strategy?
5) Are strategies implemented implicitly or explicitly?
6) Can users get advice about strategies?
7) Are selected strategies critiqued by the system?
8) If so, are they critiqued before or after sequencing?

This list of questions is not meant to be complete. It is only meant to suggest that in a computer-based authoring setting, the critical factors of an instructional design model that are expected to correlate with quality and cost effectiveness of courseware produced are those which take into account what users of various levels of experience can and actually manage to do with a particular system.

An Analogue in Word Processing

Consider the analogous situation with regard to producing computer-based written materials (i.e., word processing). There are clearly relevant levels of experience with regard to word processing, such as general knowledge of a language, typing ability, general knowledge of a computer system, and specific

experience with a particular word processing program. If it can be ascertained that users will be good typists with no prior computer experience, then ease-of-use becomes a prime factor in selecting a word processing program. However, if a requirement of the job is to produce a newsletter with pictures and text intermixed, then those requirements will be sought out, possibly sacrificing some ease-of-use.

What has happened with regard to word processing is instructive with regard to the future of automated tools for instructional design. Early word processing programs were extremely restricted text editors. These programs were line oriented -- the basic unit manipulated was a line of text and a file amounted to a collection of discrete lines of text. Next came full screen editors -- the basic unit that could be manipulated got larger as block moves and copies were possible. Then the capabilities to perform global manipulations and to merge parts of one file into another file were added. The word processors in common use on personal computers now offer additional features, including spell checking, grammatical analysis, support for tables and figures, multiple fonts, and so on.

As features are added what users actually do with a word processor undergoes subtle changes. What might have been an effective procedure for producing essays using a line editor is no longer optimal when using a full screen word processor. For example, modern word processors facilitate building an outline

first and then filling in the parts; working out of the actual line sequence with a line editor is quite difficult.

To complete this scenario, we shall contrast a a state-of-the-art desktop publishing system which can support both graphics and text in a variety of complex layouts (e.g., Aldus's PageMaker) with a prompted, style-driven document processor (e.g., Bell South's Intelligent Document Processor -- IDP). The first requires a sophisticated user. To use such a complex program effectively requires a great deal of knowledge both about the program and about how to design effective page layouts. The second kind of system enables a novice user to immediately create documents which adhere to a standard design; such a system also relieves advanced users from worrying about layout commands.

The target user population and the target problem domain is different for these systems. They should not be evaluated one against the other. Clearly there is a need for each kind of system, if market evidence is to be believed. The authoring activities in each system differ radically. In the first, users incorporate specified graphics at selected points in the text, identify fonts for text, determine page layouts, and so on. In the second, users simply respond to prompts for information and the system then creates the appropriately formatted document.

The point of this analogy is that modeling a process is not always a straightforward exercise. There are analogous extremes on the automated instructional design process spectrum. A full-featured courseware authoring environment such as TenCore or

Quest is radically different from a prompted system such as Merrill's Transaction Shells (Merrill, 1990). What users do in the first environment is radically different than what occurs in the second environment. The instructional design modeling process is intended to model what humans do with an automated system. The purpose of having a model is to determine factors that might contribute to quality of courseware or the cost effectiveness of the system. Once these factors are well-established for a specific type of system and user population, then the model may be used to predict the success of a particular instructional design effort.

The CBI Design Problem in the Air Force

The use of CBI in the Air force is expected to increase (Carter, 1990). The Air Force has extensive requirements to provide technical training to support a variety of on-going missions and weapon systems. However, resources to support this training effort are diminishing. Human resources are being lost to attrition as the Air Force cuts back on personnel. Funds to support training are also being curtailed.

As a result, there are fewer instructors and instructional designers and smaller budgets to provide the required training. The proposed solution is to make increased use of CBI. However, for this solution to be a cost-effective alternative to traditional lecture-based training, there is an additional problem to confront: there is little CBI design experience in Air Force technical training centers. What is needed, then, is

an effective CBI authoring environment that targets subject matter experts who have very little background in instructional technology or educational psychology. This requirement is evident in HQ ATC's Manpower and Training Need 89-14T: Research and Development of Computer-Based Instruction.

## Method

The general approach proposed here is to use a soft modeling approach (Falk, 1987). Soft modeling is a form of causal modeling based on the technique of partial least squares which allows researchers to make causal inferences from field data. The basic idea is that first a model which proposes causal relationships among variables is formalized. This model is then assessed against a data set or correlation matrix which expresses observed relations among measures of variables obtained in field research. Causal models typically assume that multiple independent variables influence the same dependent variable, allowing for the expression of joint causation. This soft modeling technique also allows for the possibility of intervening variables that stand between the independent/dependent variable relationships.

Additional details of the soft modeling technique are not provided here as this analysis does not involve a completed initial model. Rather, this effort is an attempt to derive an initial model to be analyzed at a later date. Another way of

expressing this methodology is to think of instructional design in tnree layers: inputs, processes, and outputs. The initial model propcsed for analysis can be encapsulated as follows:

| INPUTS | PROCESSES | OUTPUTS |
|---|---|---|
| ID experience.<br>CBI experience.<br>Subject matter experience.<br>Computer experience.<br>Instructor experience.<br>Personal data -- age, rank, sex, educ. | Times on each authoring activity.<br>Number of revisions.<br>Purpose of revisions.<br>Sequence of activities. | Overall dev. time.<br>Peer review.<br>Development cost.<br>Student results.<br>Student time under instruction. |

Table 2. Initial CBI Instructional Design Model.

Data was collected on two separate occasions to confirm an expectation of relevance for the factors indicated above. The first study involved a single subject matter expert who had no previous computer experience. He was an inst.uctor at the Air Force Academy responsible for navigation instruction. His task was to develop a CBI module on the T-37 instrument panel for use in a navigation course. Merrill's Naming Transaction Shell was the only instructional design software used in the initial study (Canfield & Spector, 1991).

A second study was conducted with eight subject matter experts at the Lowry AFB Technical Training Center and the same

instructor at the Air Force Academy. Two transaction shells were used in the follow-on study: 1) Naming the parts, and 2) Checklist procedures. Merrill's transaction shells were chosen for these studies because they appeared to provide an effective environment for a novice instructional designer and they also automated the process of delivery.

In each study, subjects had approximately 30 hours to design and develop a lesson module requiring at least one hour of student instruction. In the first study, the time was spaced out over a two week period. In the second study, the time was contained in a one week period. In both cases, total time (approximately 30 hours) included time to learn how to use the software effectively. In the first study, the subject did not create any of his own graphics, although his total time includes time to plan graphics -- what to put in each graphic, how to group parts, etc. In the second study, all but one subject also created some or all of their graphics using a simple MS-DOS based draw program, Dr. Halo. None of the subjects were acquainted with this program prior to the study.

Subjects were given an initial profile sheet to gather biographical data. There were given approximately six hours of instruction on the software. Subjects in the second study received approximately two hours of instruction on the draw program. Subjects then went about the task of planning, designing, and creating their lessons. They kept a log of observations about the software. The software kept track of the

time on each task selectable from a Transaction Shell menu. In addition, their questions and verbal observations were logged by the authors. At the end of the study, the lessons underwent a peer review procedure and subjects were debriefed and given an exit interview. Remarks were again recorded in the log.

The total development time was not allowed to exceed 30 hours because one objective of these studies was to determine if Merrill's Transaction Shells held any promise of providing a truly cost-effective courseware authoring environment. Various studies suggest that anywhere from 200 to 600 hours of total development time are required for an hour's worth of CBI (Carter, 1990). These studies would not have been possible with a full-featured authoring environment such as TenCore or Quest. Learning to use such complex software would have required far in excess of the 30 hours available.

## Results

One significant finding of these studies is that Merrill's Transaction Shells do provide a highly cost-effective authoring environment that is indeed accessible to novice CBI designers. All subjects who were able to complete the study (two of the Lowry subjects were assigned other military duties in the middle of the study) did in fact create lessons which were nearly complete enough to deliver to students. Plans are underway to test the instruction with students at both the Air Force Academy

and at Lowry AFB. The order of magnitude improvement in the development time to instruction time ratio may be slightly misleading due to the small scale of the effort, the simplicity of the lessons, and the help given in the area of graphics. However, what appears undeniably true is that even novice CBI designers can develop effective CBI for many technical training objectives in much less time than possible using existing courseware authoring environments.

With regard to the proposed initial CBI design model (Table 2 above), all of the proposed factors could in fact be observed or measured. There are as yet not enough subjects on which to base a generalization, much less a partial least squares analysis. However, some specific observations are in order, as they reflect unexpected findings. One of the Lowry subjects had some CBI experience, extensive experience in the subject area, and some experience with computers. Another subject had no CBI experience but recent experience with the computers used in this study (Zenith Z-248s) and a computer at home. The individual with recent Z-248 experience was able to create two lesson modules, whereas all others completed the equivalent of a single lesson module. Lesson modules represented between one and three hours of instruction for a student. This suggests that recent computer experience might be more relevant to CBI design success using Merrill's Transaction Shells than CBI design experience in another environment. It also suggests that an additional input factor might be whether or not the individual owns a personal

computer and, if so, what type of computer is owned and how it is used. This data will be collected in future studies.

Another observation is that the pattern of revisions is not at all constant, yet all subjects produced roughly equivalent results. For example, some subjects preferred to postpone all revisions until a complete version of the lesson had been created. Others revised each part of the lesson as they proceeded through the process. Peer review of these lessons revealed no significant differences in quality. As yet there is no emergent pattern between input profiles and revision profiles.

Of the nine subjects involved in these studies, only one had any significant CBI experience -- about a year's experience authoring in the Quest environment. This individual gave the lowest estimate of the time that would have been required to achieve a comparable lesson in a full-featured CBI authoring environment. All but one had extensive experience as subject matter experts (SMEs). Even the relatively inexperienced SME was able to complete a lesson module, although he spent more off-line time planning the lesson than the others. Only two of the nine had extensive experience as instructors. One of these indicated on the first day of the study that he thought that stand-up, lecture-based instruction was the only effective way to teach. At the end of the week, he had become an outspoken advocate of CBI.

Five of the seven subjects who completed the study indicated a desire to deliver their lessons to students and a willingness

to continue to use this form of CBI. All five of those who tried to use the Checklist Procedure authoring software found it problematic -- generally less understandable than the Naming software. All found some faults with the software -- lack of integrated functionalities, limited word processing and graphics capabilities, too much student control allowed, etc. However, all indicated that this software was effective and contributed to their productivity.

One last observation is that none of the subjects ever expressed a fear of being replaced by the computer. This question was posed directly only to the Air Force Academy subject. His response was that he viewed the software as yet another resource or support tool. This opinion was expressed indirectly by all of the Lowry AFB subjects.

## Conclusions

There is an obvious need to continue with these studies. Additional subjects and student data are needed before a complete analysis can be performed. Follow-on studies are being planned for Lowry AFB and the Air Force Academy as well as other sites.

There appears to be good potential for Merrill's Transaction Shell approach to courseware authoring. This approach forms the core of the Advanced Instructional Design Advisor (AIDA) now being developed by the Armstrong Laboratory as an experimental research platform (Spector, 1990).

However, other approaches should also be studied and alternative models developed. Duchastel (1990) contrasts instructional design expert systems (Merrill's ID Expert) with critiquing systems (his proposed ID Workbench). Expert systems are faulted for being too restrictive or unpalatable for experienced users. While Duchastel's criticism of expert systems for instructional design is somewhat overstated, our argument supports his general view that different kinds of authoring environments are appropriate for different kinds of authors. Our view in fact goes further in that it hypothesizes different underlying models and human activities for the experienced and the inexperienced.

Duchastel mentions a third kind of possibility for an automated instructional design system -- an intelligent tutoring system. Tennyson also proposed the possibility of an ITS for instructional design (Tennyson, 1990). There are still other possibilities. Progressive Learning Systems is developing a system called ID Advisor for the Armstrong Laboratory as part of a Phase II Small Business Innovative Research proposal. ID Advisor will be a case-based instructional design advisor, rather than a rule-based system of the type criticized by Duchastel. Robert Gagné is now a Senior Research Fellow at the Armstrong Laboratory and is pursuing a dialogue-based guided approach to instructional design. As these various alternatives become implemented as software prototypes, appropriate underlying models will be postulated and evaluated.

In conclusion, we can speculate that what has happened in the domain of word processing is happening in the domain of instructional design. Systems are growing in power and complexity. Soon the entire spectrum from the full-featured courseware authoring workbench to the prompted fill-in-the-blanks courseware automaton will be well represented in the market place. What will then be needed are useful predictive models to aid in the selection and use of the appropriate tools and environments.

## References


Andrews, D. H. & Goodson, L. A. (1980). A comparative analysis of models of instructional design. Journal of Instructional Development, 3(4), 2-16.

Carter, J. (1990). The interactive courseware decision handbook. Randolph AFB, TX: HQ ATC Technical Report.

Canfield, A. M. & Spector, J. M. (1991). A Pilot Study of the Naming Transaction Shell (AFHRL TP-91-xx). Technical paper for the Training Systems Division of the Air Force Human Resources Laboratory.

Dick, W. & Carey, L. (1985). The Systematic Design of Instruction. Glenview, IL: Scott Foresman.

Duchastel, P. C. (1990). Cognitive designs for instructional design. Instructional Science, 19(6), 437-444.

Falk, R. F. (1987). A primer for soft modeling. Berkeley, CA: University of California, The Institute for Human Development.

Gagné, R. M., Briggs, L. J., & Wager, W. W. (1988). Principles of Instructional Design (3rd edition). Fort Worth, TX: Holt, Rinehart, and Winston.

Geol, V. and Pirolli, P. (1989). Motivating the notion of generic design within information processing: The design space problem. AI Magazine, 10 (1), 18-36.

Glaser, R. (1989). Expertise and learning: How do we think about instructional processes now that we have discovered knowledge structures? In D. Klahr and K. Kotovsky (eds.), Complex Information Processing: The Impact of Herbert Simon, Hillsdale, NJ: Erlbaum.

Merrill, M. D. Li, Z., & Jones, M. K. (1990). Second generation instructional design (ID-2). Educational Technology, 30(2).

Pirolli, P. (1989). On the art of building: Putting a new instructional design into practice. In H. Burns and J. Parlette (eds.), Proceedings of the 2nd Intelligent Tutoring Systems Research Forum, San Antonio, TX, April, 1989.

Spector, J. M. (1990). Designing and Developing an Advanced Instructional Design Advisor (AFHRL-TP-90-52). Brooks AFB, TX: Technical Paper for the Training Systems Division of the Air Force Human Resources Laboratory.

Spector, J. M., Muraida, D. J., & Dallman, B. E. (1990). Establishing instructional strategies for advanced interactive technologies. Proceedings of the Psychology in the DoD Symposium, USAFA, CO, April, 1990.

Tennyson, R. D. (1989). Cognitive science update of instructional systems design models. Brooks AFB, TX: Technical presentation for the AIDA project at the Air Force Armstrong Laboratory (AL/HRTC, formerly AFHRL/IDC).

Tennyson, R. D. (1990). Framework specifications document for an instructional systems development expert system. Brooks AFB, TX: Technical presentation for the AIDA project at the Air Force Armstrong Laboratory (AL/HRTC, formerly AFHRL/IDC).